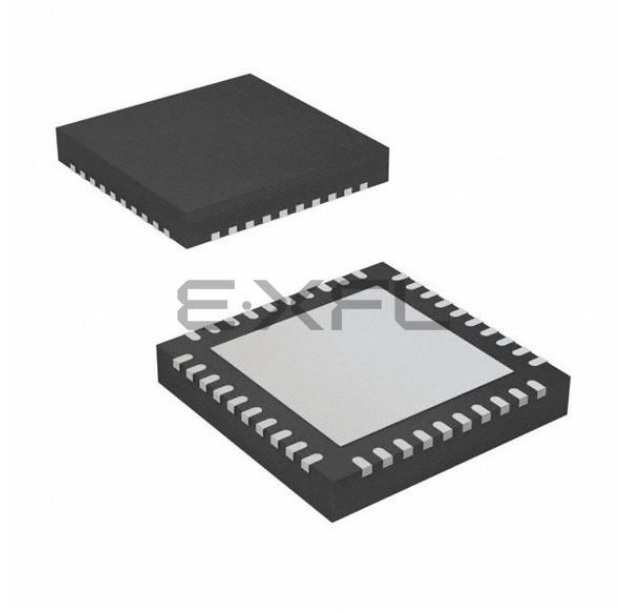


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	64MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	36
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 35x10b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	40-UFQFN Exposed Pad
Supplier Device Package	40-UQFN (5x5)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f45k40-e-mv

1.0 DEVICE OVERVIEW

This document contains device specific information for the following devices:

- PIC18F26K40
- PIC18LF26K40
- PIC18F45K40
- PIC18LF45K40
- PIC18F46K40
- PIC18LF46K40

This family offers the advantages of all PIC18 microcontrollers – namely, high computational performance at an economical price – with the addition of high-endurance, Program Flash Memory. In addition to these features, the PIC18(L)F2x/4xK40 family introduces design enhancements that make these microcontrollers a logical choice for many high-performance, power sensitive applications.

1.1 New Core Features

1.1.1 XLP TECHNOLOGY

All of the devices in the PIC18(L)F2x/4xK40 family incorporate a range of features that can significantly reduce power consumption during operation. Key items include:

- **Alternate Run Modes:** By clocking the controller from the secondary oscillator or the internal oscillator block, power consumption during code execution can be reduced by as much as 90%.
- **Multiple Idle Modes:** The controller can also run with its CPU core disabled but the peripherals still active. In these states, power consumption can be reduced even further, to as little as 4% of normal operation requirements.
- **On-the-fly Mode Switching:** The power-managed modes are invoked by user code during operation, allowing the user to incorporate power-saving ideas into their application's software design.
- **Peripheral Module Disable:** Modules that are not being used in the code can be selectively disabled using the PMD module. This further reduces the power consumption.

1.1.2 MULTIPLE OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18(L)F2x/4xK40 family offer several different oscillator options. The PIC18(L)F2x/4xK40 family can be clocked from several different sources:

- HFINTOSC
  - 1-64 MHz precision digitally controlled internal oscillator
- LFINTOSC
  - 31 kHz internal oscillator
- EXTOSC
  - External clock (EC)
  - Low-power oscillator (LP)
  - Medium power oscillator (XT)
  - High-power oscillator (HS)
- SOSC
  - Secondary oscillator circuit operating at 31 kHz
- A Phase Lock Loop (PLL) frequency multiplier (4x) is available to the External Oscillator modes enabling clock speeds of up to 64 MHz
- Fail-Safe Clock Monitor: This option constantly monitors the main clock source against a reference signal provided by the LFINTOSC. If a clock failure occurs, the controller is switched to the internal oscillator block, allowing for continued operation or a safe application shutdown.

3.0 DEVICE CONFIGURATION

Device configuration consists of Configuration Words, Code Protection, Device ID and Rev ID.

3.1 Configuration Words

There are six Configuration Word bits that allow the user to setup the device with several choices of oscillators, Resets and memory protection options. These are implemented as Configuration Word 1 through Configuration Word 6 at 300000h through 30000Bh.

Note:	The $\overline{\text{DEBUG}}$ bit in Configuration Words is managed automatically by device development tools including debuggers and programmers. For normal device operation, this bit should be maintained as a '1'.

REGISTER 3-13: REVISION ID: REVISION ID REGISTER

R	R	R	R	R	R	R	R
1	0	1	0	MJRREV<5:2>			
bit 15							bit 8

R	R	R	R	R	R	R	R
MJRREV<1:0>		MNRREV<5:0>					
bit 7							bit 0

Legend:			
R = Readable bit	'1' = Bit is set	0' = Bit is cleared	x = Bit is unknown

bit 15-12 **Read as '1010'**
These bits are fixed with value '1010' for all devices in this family.

bit 11-6 **MJRREV<5:0>:** Major Revision ID bits
These bits are used to identify a major revision. A major revision is indicated by an all-layer revision (A0, B0, C0, etc.).
Revision A = 6'b00_0001

bit 5-0 **MNRREV<5:0>:** Minor Revision ID bits
These bits are used to identify a minor revision.

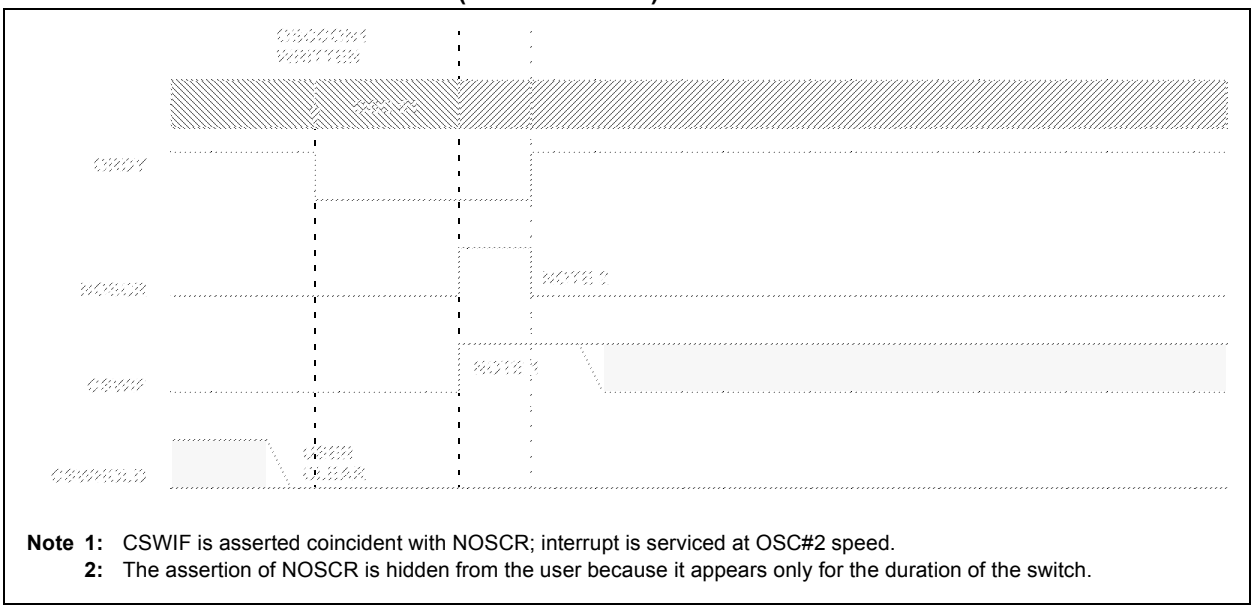
4.4.2 CLOCK SWITCH AND SLEEP

If OSCCON1 is written with a new value and the device is put to Sleep before the switch completes, the switch will not take place and the device will enter Sleep mode.

When the device wakes from Sleep and the CSWHOLD bit is clear, the device will wake with the 'new' clock active, and the clock switch interrupt flag bit (CSWIF) will be set.

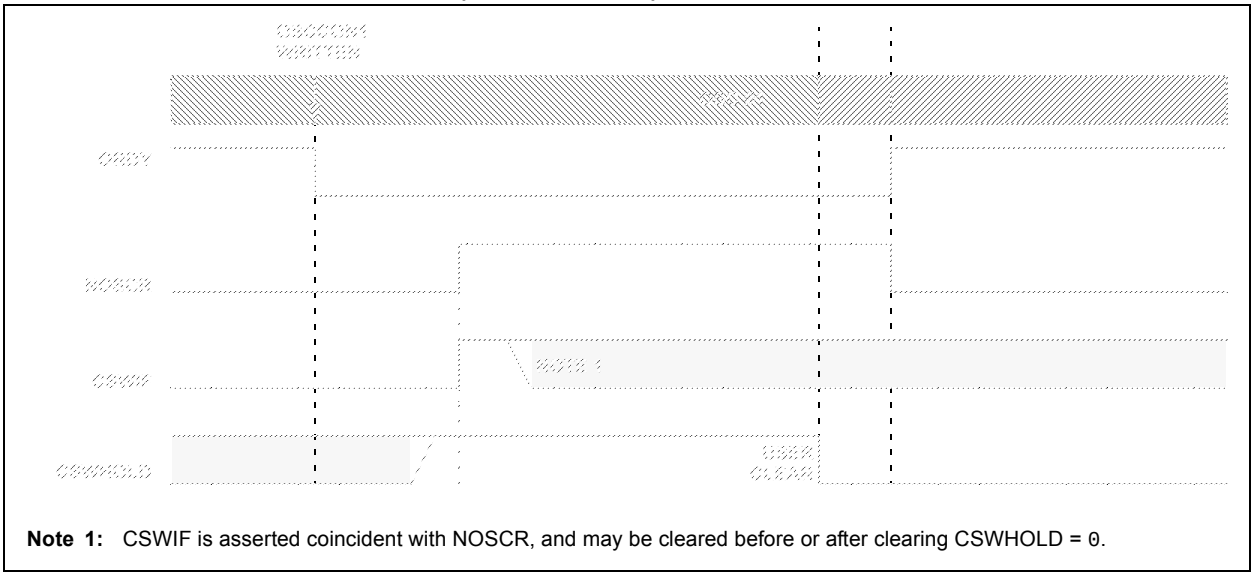
When the device wakes from Sleep and the CSWHOLD bit is set, the device will wake with the 'old' clock active and the new clock will be requested again.

FIGURE 4-6: CLOCK SWITCH (CSWHOLD = 0)



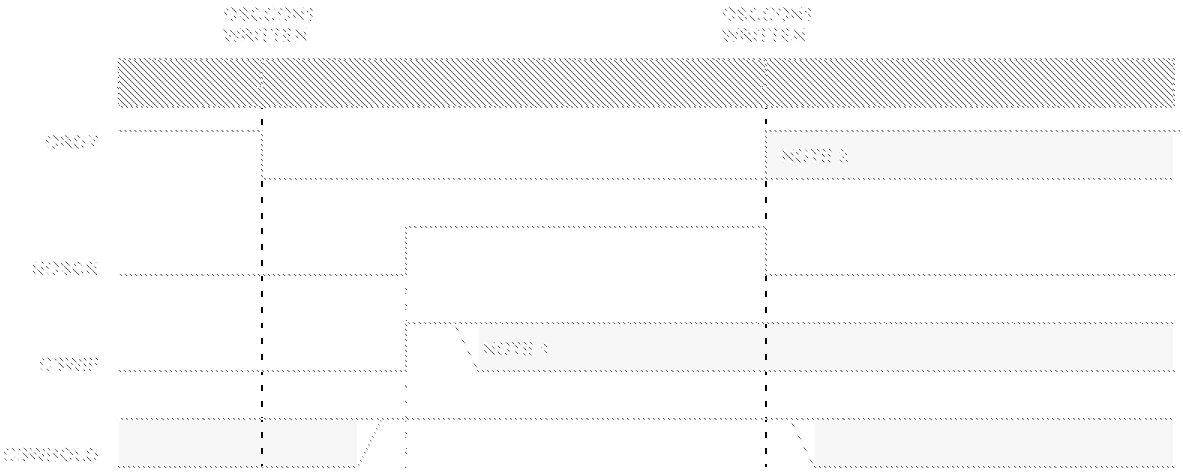
Note 1: CSWIF is asserted coincident with NOSCR; interrupt is serviced at OSC#2 speed.
2: The assertion of NOSCR is hidden from the user because it appears only for the duration of the switch.

FIGURE 4-7: CLOCK SWITCH (CSWHOLD = 1)



Note 1: CSWIF is asserted coincident with NOSCR, and may be cleared before or after clearing CSWHOLD = 0.

FIGURE 4-8: CLOCK SWITCH ABANDONED



Note 1: CSWIF may be cleared before or after rewriting OSCCON1; CSWIF is not automatically cleared.
2: ORDY = 0 if OSCCON1 does not match OSCCON2; a new switch will begin.

6.4 Register Definitions: Voltage Regulator Control

REGISTER 6-1: VREGCON: VOLTAGE REGULATOR CONTROL REGISTER(1)

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-1/1
—	—	—	—	—	—	VREGPM	Reserved
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-2 **Unimplemented:** Read as '0'

bit 1 **VREGPM:** Voltage Regulator Power Mode Selection bit
- 1 = Low-Power Sleep mode enabled in Sleep(2)
  Draws lowest current in Sleep, slower wake-up
- 0 = Normal Power mode enabled in Sleep(2)
  Draws higher current in Sleep, faster wake-up

bit 0 **Reserved:** Read as '1'. Maintain this bit set.

Note 1: PIC18F2x/4xK40 only.
2: See **Section 37.0 "Electrical Specifications"**.

9.1 Register Definitions: Windowed Watchdog Timer Control

REGISTER 9-1: WDTCON0: WATCHDOG TIMER CONTROL REGISTER 0

U-0	U-0	R/W(3)-q/q(2)	R/W(3)-q/q(2)	R/W(3)-q/q(2)	R/W(3)-q/q(2)	R/W(3)-q/q(2)	R/W-0/0
—	—	WDTPS<4:0>					SEN
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	q = Value depends on condition

bit 7-6 **Unimplemented:** Read as '0'

bit 5-1 **WDTPS<4:0>:** Watchdog Timer Prescale Select bits(1)

Bit Value = Prescale Rate

11111 = Reserved. Results in minimum interval (1:32)

•

•

•

10011 = Reserved. Results in minimum interval (1:32)

10010 = 1:8388608 (2^{23}) (Interval 256s nominal)
10001 = 1:4194304 (2^{22}) (Interval 128s nominal)
10000 = 1:2097152 (2^{21}) (Interval 64s nominal)
01111 = 1:1048576 (2^{20}) (Interval 32s nominal)
01110 = 1:524288 (2^{19}) (Interval 16s nominal)
01101 = 1:262144 (2^{18}) (Interval 8s nominal)
01100 = 1:131072 (2^{17}) (Interval 4s nominal)
01011 = 1:65536 (Interval 2s nominal) (Reset value)
01010 = 1:32768 (Interval 1s nominal)
01001 = 1:16384 (Interval 512 ms nominal)
01000 = 1:8192 (Interval 256 ms nominal)
00111 = 1:4096 (Interval 128 ms nominal)
00110 = 1:2048 (Interval 64 ms nominal)
00101 = 1:1024 (Interval 32 ms nominal)
00100 = 1:512 (Interval 16 ms nominal)
00011 = 1:256 (Interval 8 ms nominal)
00010 = 1:128 (Interval 4 ms nominal)
00001 = 1:64 (Interval 2 ms nominal)
00000 = 1:32 (Interval 1 ms nominal)

bit 0 **SEN:** Software Enable/Disable for Watchdog Timer bit

If WDTE<1:0> = 1x:
This bit is ignored.
If WDTE<1:0> = 01:
1 = WDT is turned on
0 = WDT is turned off
If WDTE<1:0> = 00:
This bit is ignored.

Note 1: Times are approximate. WDT time is based on 31 kHz LFINTOSC.

2: When WDTCPS <4:0> in CONFIG3L = 11111, the Reset value of WDTPS<4:0> is 01011. Otherwise, the Reset value of WDTPS<4:0> is equal to WDTCPS<4:0> in CONFIG3L.

3: When WDTCPS <4:0> in CONFIG3L ≠ 11111, these bits are read-only.

19.2 Timer1/3/5 Operation

The Timer1/3/5 module is a 16-bit incrementing counter which is accessed through the TMRxH:TMRxL register pair. Writes to TMRxH or TMRxL directly update the counter.

When used with an internal clock source, the module is a timer and increments on every instruction cycle. When used with an external clock source, the module can be used as either a timer or counter and increments on every selected edge of the external source.

Timer1/3/5 is enabled by configuring the ON and GE bits in the TxCON and TxGCON registers, respectively. Table 19-2 displays the Timer1/3/5 enable selections.

TABLE 19-2: TIMER1/3/5 ENABLE SELECTIONS

ON	GE	Timer1/3/5 Operation
1	1	Count Enabled
1	0	Always On
0	1	Off
0	0	Off

19.3 Clock Source Selection

The CS<3:0> bits of the TMRxCLK register (Register 19-3) are used to select the clock source for Timer1/3/5. The four TMRxCLK bits allow the selection of several possible synchronous and asynchronous clock sources. Register 19-3 displays the clock source selections.

19.3.1 INTERNAL CLOCK SOURCE

When the internal clock source is selected the TMRxH:TMRxL register pair will increment on multiples of FOSC as determined by the Timer1/3/5 prescaler.

When the FOSC internal clock source is selected, the Timer1/3/5 register value will increment by four counts every instruction clock cycle. Due to this condition, a 2 LSB error in resolution will occur when reading the Timer1/3/5 value. To utilize the full resolution of Timer1/3/5, an asynchronous input signal must be used to gate the Timer1/3/5 clock input.

The following asynchronous sources may be used at the Timer1/3/5 gate:

- Asynchronous event on the TxGPPS pin
- TMR0OUT
- TMR1/3/5OUT (excluding the TMR for which it is being used)
- TMR 2/4/6OUT (post-scaled)
- CCP1/2OUT
- PWM3/4OUT
- CMP1/2OUT
- ZCDOUT

Note: In Counter mode, a falling edge must be registered by the counter prior to the first incrementing rising edge after any one or more of the following conditions:

- Timer1/3/5 enabled after POR
- Write to TMRxH or TMRxL
- Timer1/3/5 is disabled
- Timer1/3/5 is disabled (TMRxON = 0) when TxCKI is high then Timer1/3/5 is enabled (TMRxON = 1) when TxCKI is low.

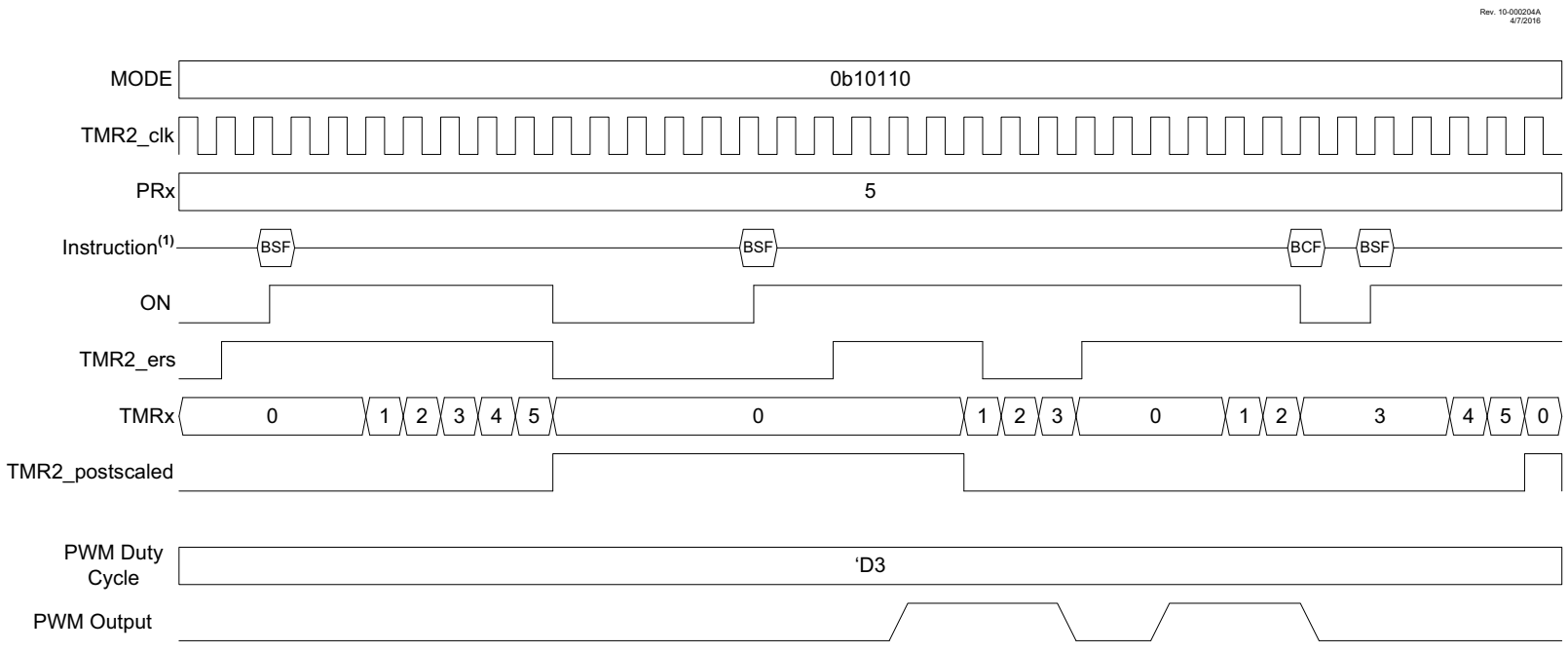
19.3.2 EXTERNAL CLOCK SOURCE

When the external clock source is selected, the Timer1/3/5 module may work as a timer or a counter.

When enabled to count, Timer1/3/5 is incremented on the rising edge of the external clock input of the TxCKIPPS pin. This external clock source can be synchronized to the microcontroller system clock or it can run asynchronously.

When used as a timer with a clock oscillator, an external 32.768 kHz crystal can be used in conjunction with the dedicated secondary internal oscillator circuit.

FIGURE 20-13: LEVEL-TRIGGERED HARDWARE LIMIT ONE-SHOT MODE TIMING DIAGRAM (MODE = 10110)



Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

REGISTER 21-1: CCPxCON: CCPx CONTROL REGISTER (CONTINUED)

bit 3-0 **MODE<3:0>:** CCPx Mode Select bits

MODE	Operating Mode	Operation	Set CCPxIF
11xx	PWM	PWM operation	Yes
1011	Compare	Pulse output; clear TMR1(2)	Yes
1010		Pulse output	Yes
1001		Clear output(1)	Yes
1000		Set output(1)	Yes
0111	Capture	Every 16th rising edge of CCPx input	Yes
0110		Every 4th rising edge of CCPx input	Yes
0101		Every rising edge of CCPx input	Yes
0100		Every falling edge of CCPx input	Yes
0011		Every edge of CCPx input	Yes
0010	Compare	Toggle output	Yes
0001		Toggle output; clear TMR1(2)	Yes
0000	Disabled		—

Note 1: The set and clear operations of the Compare mode are reset by setting MODE = 4'b0000 or EN = 0.

2: When MODE = 0001 or 1011, then the timer associated with the CCP module is cleared. TMR1 is the default selection for the CCP module, so it is used for indication purpose only.

REGISTER 21-2: CCPTMRS: CCP TIMERS CONTROL REGISTER

R/W-0/0	R/W-1/1	R/W-0/0	R/W-1/1	R/W-0/0	R/W-1/1	R/W-0/0	R/W-1/1
P4TSEL<1:0>		P3TSEL<1:0>		C2TSEL<1:0>		C1TSEL<1:0>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-6 **P4TSEL<1:0>:** PWM4 Timer Selection bits
11 = PWM4 based on TMR6
10 = PWM4 based on TMR4
01 = PWM4 based on TMR2
00 = Reserved

bit 5-4 **P3TSEL<1:0>:** PWM3 Timer Selection bits
11 = PWM3 based on TMR6
10 = PWM3 based on TMR4
01 = PWM3 based on TMR2
00 = Reserved

bit 3-2 **C2TSEL<1:0>:** CCP2 Timer Selection bits
11 = CCP2 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode
10 = CCP2 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode
01 = CCP2 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode
00 = Reserved

bit 1-0 **C1TSEL<1:0>:** CCP1 Timer Selection bits
11 = CCP1 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode
10 = CCP1 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode
01 = CCP1 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode
00 = Reserved

FIGURE 26-20: I²C SLAVE, 10-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 0, DHEN = 0)

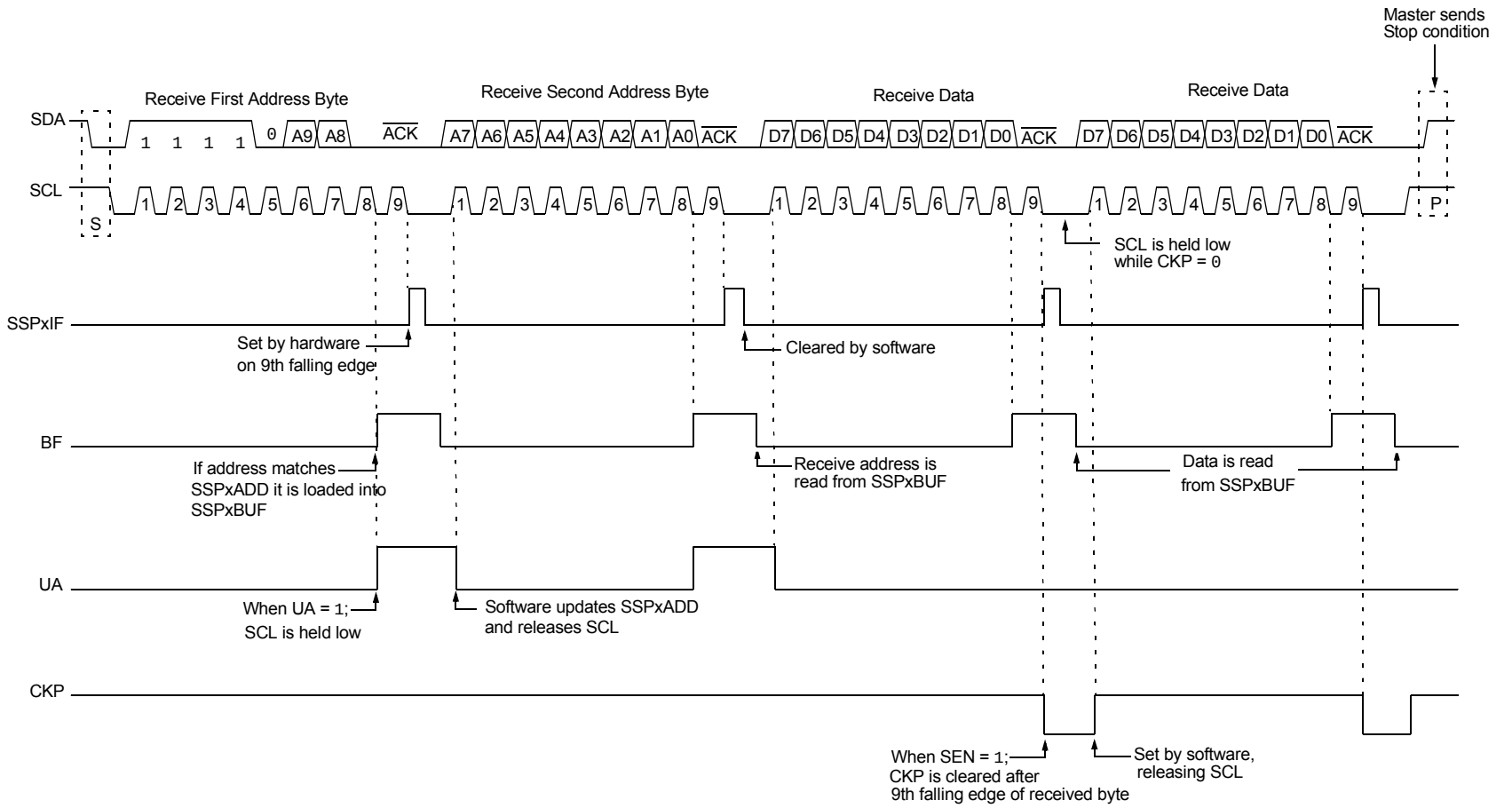
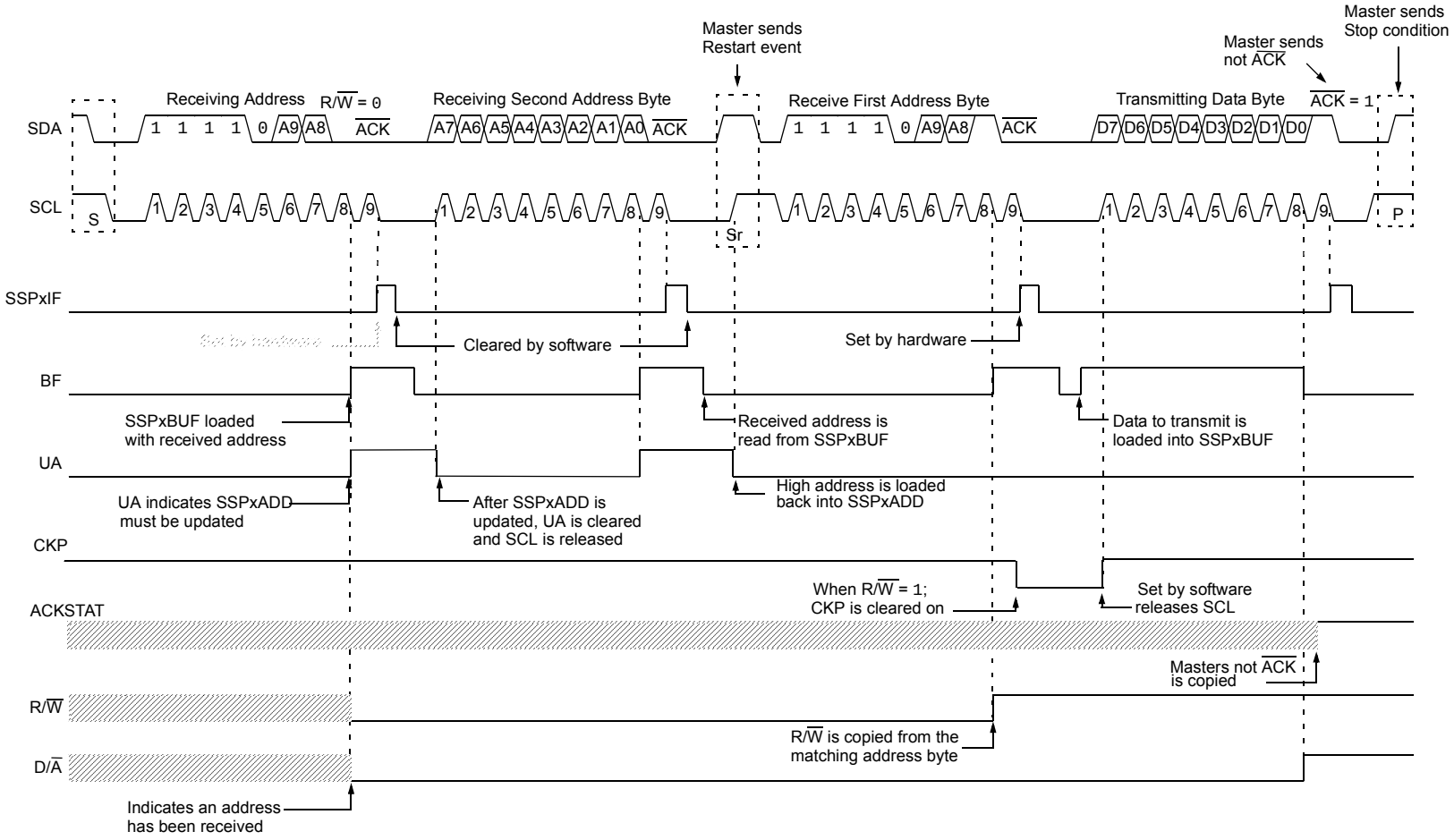


FIGURE 26-22: I²C SLAVE, 10-BIT ADDRESS, TRANSMISSION (SEN = 0, AHEN = 0, DHEN = 0)



27.2 EUSART Asynchronous Mode

The EUSART transmits and receives data using the standard non-return-to-zero (NRZ) format. NRZ is implemented with two levels: a VOH Mark state which represents a '1' data bit, and a VOL Space state which represents a '0' data bit. NRZ refers to the fact that consecutively transmitted data bits of the same value stay at the output level of that bit without returning to a neutral level between each bit transmission. An NRZ transmission port idles in the Mark state. Each character transmission consists of one Start bit followed by eight or nine data bits and is always terminated by one or more Stop bits. The Start bit is always a space and the Stop bits are always marks. The most common data format is eight bits. Each transmitted bit persists for a period of 1/(Baud Rate). An on-chip dedicated 8-bit/16-bit Baud Rate Generator is used to derive standard baud rate frequencies from the system oscillator. See Table 27-5 for examples of baud rate configurations.

The EUSART transmits and receives the LSb first. The EUSART's transmitter and receiver are functionally independent, but share the same data format and baud rate. Parity is not supported by the hardware, but can be implemented in software and stored as the ninth data bit.

27.2.1 EUSART ASYNCHRONOUS TRANSMITTER

The EUSART transmitter block diagram is shown in Figure 27-1. The heart of the transmitter is the serial Transmit Shift Register (TSR), which is not directly accessible by software. The TSR obtains its data from the transmit buffer, which is the TXxREG register.

27.2.1.1 Enabling the Transmitter

The EUSART transmitter is enabled for asynchronous operations by configuring the following three control bits:

- TXEN = 1
- SYNC = 0
- SPEN = 1

All other EUSART control bits are assumed to be in their default state.

Setting the TXEN bit of the TXxSTA register enables the transmitter circuitry of the EUSART. Clearing the SYNC bit of the TXxSTA register configures the EUSART for asynchronous operation. Setting the SPEN bit of the RCxSTA register enables the EUSART and automatically configures the TXx/CKx I/O pin as an output. If the TXx/CKx pin is shared with an analog peripheral, the analog I/O function must be disabled by clearing the corresponding ANSEL bit.

Note:	The TXxIF Transmitter Interrupt flag is set when the TXEN enable bit is set.

27.2.1.2 Transmitting Data

A transmission is initiated by writing a character to the TXxREG register. If this is the first character, or the previous character has been completely flushed from the TSR, the data in the TXxREG is immediately transferred to the TSR register. If the TSR still contains all or part of a previous character, the new character data is held in the TXxREG until the Stop bit of the previous character has been transmitted. The pending character in the TXxREG is then transferred to the TSR in one TCY immediately following the Stop bit transmission. The transmission of the Start bit, data bits and Stop bit sequence commences immediately following the transfer of the data to the TSR from the TXxREG.

27.2.1.3 Transmit Data Polarity

The polarity of the transmit data can be controlled with the SCKP bit of the BAUDxCON register. The default state of this bit is '0' which selects high true transmit idle and data bits. Setting the SCKP bit to '1' will invert the transmit data resulting in low true idle and data bits. The SCKP bit controls transmit data polarity in Asynchronous mode only. In Synchronous mode, the SCKP bit has a different function. See **Section 27.5.1.2 "Clock Polarity"**.

27.2.1.4 Transmit Interrupt Flag

The TXxIF interrupt flag bit of the PIR3 register is set whenever the EUSART transmitter is enabled and no character is being held for transmission in the TXxREG. In other words, the TXxIF bit is only clear when the TSR is busy with a character and a new character has been queued for transmission in the TXxREG. The TXxIF flag bit is not cleared immediately upon writing TXxREG. TXxIF becomes valid in the second instruction cycle following the write execution. Polling TXxIF immediately following the TXxREG write will return invalid results. The TXxIF bit is read-only, it cannot be set or cleared by software.

The TXxIF interrupt can be enabled by setting the TXxIE interrupt enable bit of the PIE3 register. However, the TXxIF flag bit will be set whenever the TXxREG is empty, regardless of the state of TXxIE enable bit.

To use interrupts when transmitting data, set the TXxIE bit only when there is more data to send. Clear the TXxIE interrupt enable bit upon writing the last character of the transmission to the TXxREG.

27.2.2.8 Asynchronous Reception Setup:

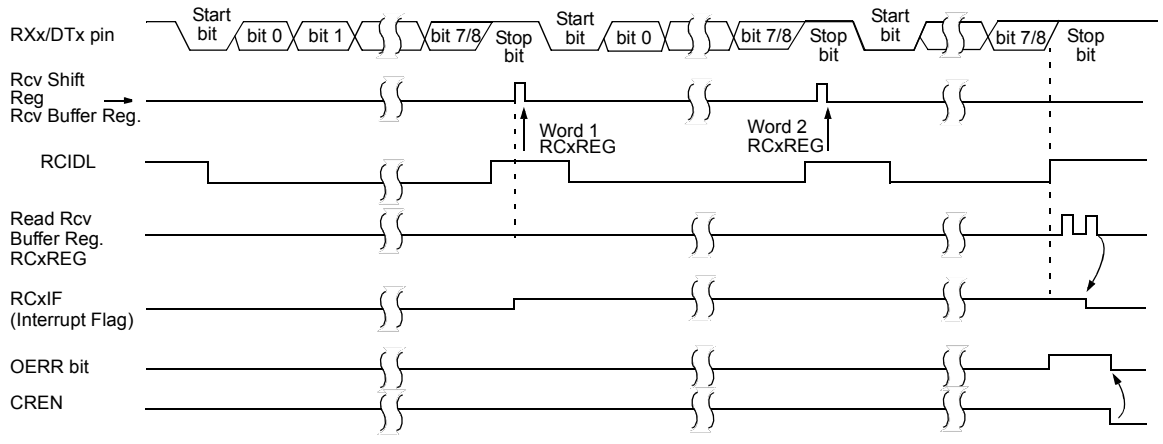
1. Initialize the SPxBRGH:SPxBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 27.4 "EUSART Baud Rate Generator (BRG)"**).
2. Clear the ANSEL bit for the RXx pin (if applicable).
3. Enable the serial port by setting the SPEN bit. The SYNC bit must be clear for asynchronous operation.
4. If interrupts are desired, set the RCxIE bit of the PIE3 register and the GIE and PEIE bits of the INTCON register.
5. If 9-bit reception is desired, set the RX9 bit.
6. Enable reception by setting the CREN bit.
7. The RCxIF interrupt flag bit will be set when a character is transferred from the RSR to the receive buffer. An interrupt will be generated if the RCxIE interrupt enable bit was also set.
8. Read the RCxSTA register to get the error flags and, if 9-bit data reception is enabled, the ninth data bit.
9. Get the received eight Least Significant data bits from the receive buffer by reading the RCxREG register.
10. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.

27.2.2.9 9-Bit Address Detection Mode Setup

This mode would typically be used in RS-485 systems. To set up an Asynchronous Reception with Address Detect Enable:

1. Initialize the SPxBRGH:SPxBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 27.4 "EUSART Baud Rate Generator (BRG)"**).
2. Clear the ANSEL bit for the RXx pin (if applicable).
3. Enable the serial port by setting the SPEN bit. The SYNC bit must be clear for asynchronous operation.
4. If interrupts are desired, set the RCxIE bit of the PIE3 register and the GIE and PEIE bits of the INTCON register.
5. Enable 9-bit reception by setting the RX9 bit.
6. Enable address detection by setting the ADDEN bit.
7. Enable reception by setting the CREN bit.
8. The RCxIF interrupt flag bit will be set when a character with the ninth bit set is transferred from the RSR to the receive buffer. An interrupt will be generated if the RCxIE interrupt enable bit was also set.
9. Read the RCxSTA register to get the error flags. The ninth data bit will always be set.
10. Get the received eight Least Significant data bits from the receive buffer by reading the RCxREG register. Software determines if this is the device's address.
11. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.
12. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and generate interrupts.

FIGURE 27-5: ASYNCHRONOUS RECEPTION



Note: This timing diagram shows three words appearing on the RXx input. The RCxREG (receive buffer) is read after the third word, causing the OERR (overrun) bit to be set.

REGISTER 31-11: ADCAP: ADC ADDITIONAL SAMPLE CAPACITOR SELECTION REGISTER

U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	ADCAP<4:0>				
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-5 **Unimplemented**: Read as ‘0’

bit 4-0 **ADCAP<4:0>**: ADC Additional Sample Capacitor Selection bits
11111 = 31 pF
11110 = 30 pF
11101 = 29 pF
•
•
•
00011 = 3 pF
00010 = 2 pF
00001 = 1 pF
00000 = No additional capacitance

REGISTER 31-12: ADRPT: ADC REPEAT SETTING REGISTER

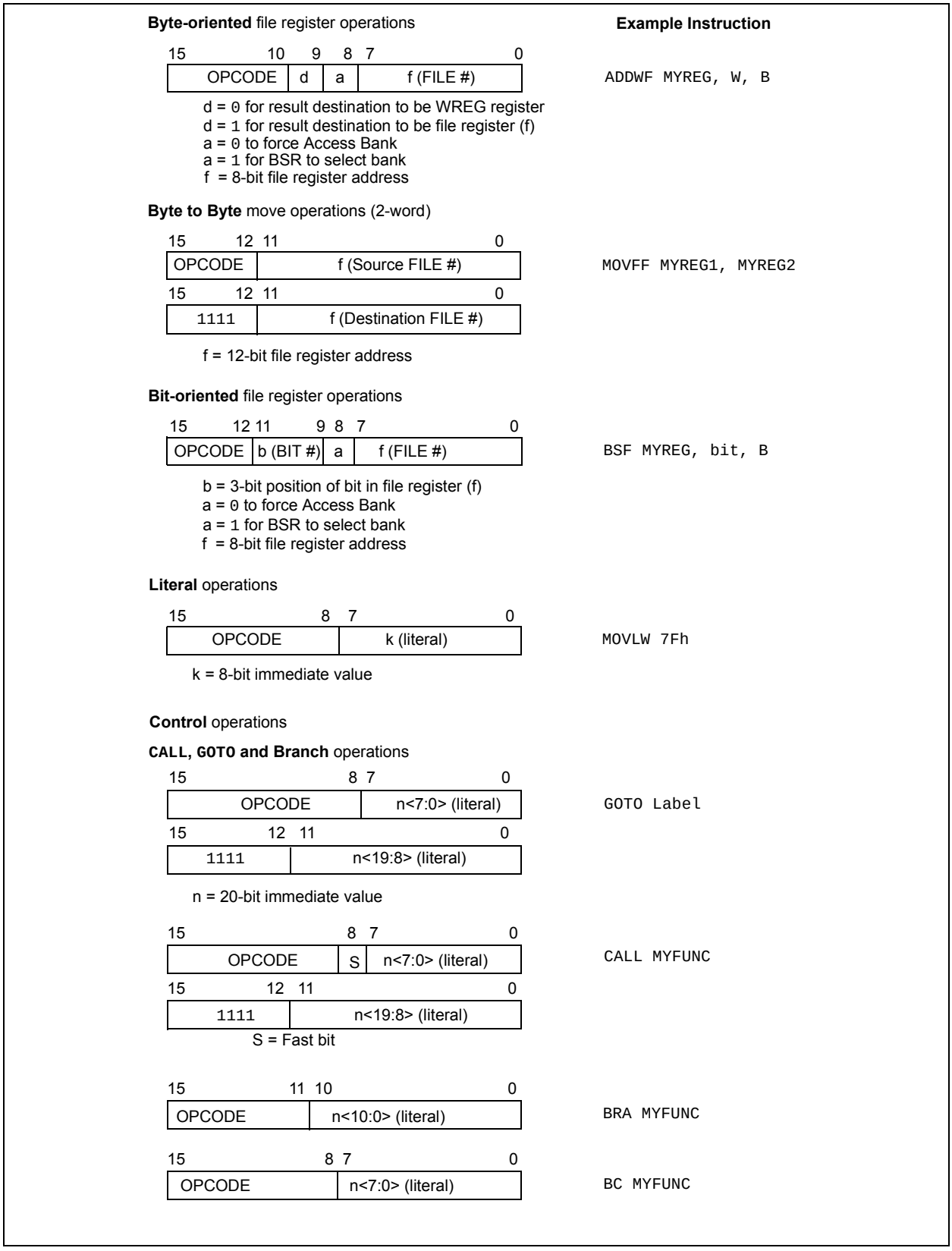
R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
ADRPT<7:0>							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-0 **Unimplemented**: Read as ‘0’

bit 7-0 **ADRPT<7:0>**: ADC Repeat Threshold bits
Counts the number of times that the ADC has been triggered and is used along with ADCNT to determine when the error threshold is checked when the computation is Low-pass Filter, Burst Average, or Average modes. See Table 31-3 for more details.

FIGURE 35-1: General Format for Instructions



TBLWT **Table Write**

Syntax: TBLWT (*; *+; *-; +*)

Operands: None

Operation: if TBLWT*,
(TABLAT) → Holding Register;
TBLPTR – No Change;
if TBLWT*+,
(TABLAT) → Holding Register;
(TBLPTR) + 1 → TBLPTR;
if TBLWT*-,
(TABLAT) → Holding Register;
(TBLPTR) – 1 → TBLPTR;
if TBLWT+*,
(TBLPTR) + 1 → TBLPTR;
(TABLAT) → Holding Register;

Status Affected: None

Encoding:

0000	0000	0000	11nn nn=0 * =1 *+ =2 *- =3 +*

Description: This instruction uses the three LSBs of TBLPTR to determine which of the eight holding registers the TABLAT is written to. The holding registers are used to program the contents of Program Memory (P.M.). (Refer to **Section 11.1 "Program Flash Memory"** for additional details on programming Flash memory.)
The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-MByte address range. The LSb of the TBLPTR selects which byte of the program memory location to access.

TBLPTR[0] = 0: Least Significant Byte of Program Memory Word
TBLPTR[0] = 1: Most Significant Byte of Program Memory Word

The TBLWT instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Read TABLAT)	No operation	No operation (Write to Holding Register)

TBLWT **Table Write (Continued)**

Example1:

```
TBLWT *+;
```

Before Instruction
TABLAT = 55h
TBLPTR = 00A356h
HOLDING REGISTER (00A356h) = FFh
After Instructions (table write completion)
TABLAT = 55h
TBLPTR = 00A357h
HOLDING REGISTER (00A356h) = 55h

Example 2:

```
TBLWT +*;
```

Before Instruction
TABLAT = 34h
TBLPTR = 01389Ah
HOLDING REGISTER (01389Ah) = FFh
HOLDING REGISTER (01389Bh) = FFh
After Instruction (table write completion)
TABLAT = 34h
TBLPTR = 01389Bh
HOLDING REGISTER (01389Ah) = FFh
HOLDING REGISTER (01389Bh) = 34h

35.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, PIC18(L)F2x/4xK40 devices also provide an optional extension to the core CPU functionality. The added features include eight additional instructions that augment indirect and indexed addressing operations and the implementation of Indexed Literal Offset Addressing mode for many of the standard PIC18 instructions.

The additional features of the extended instruction set are disabled by default. To enable them, users must set the XINST Configuration bit.

The instructions in the extended set can all be classified as literal operations, which either manipulate the File Select Registers, or use them for indexed addressing. Two of the instructions, ADDFSR and SUBFSR, each have an additional special instantiation for using FSR2. These versions (ADDULNK and SUBULNK) allow for automatic return after execution.

The extended instructions are specifically implemented to optimize re-entrant program code (that is, code that is recursive or that uses a software stack) written in high-level languages, particularly C. Among other things, they allow users working in high-level languages to perform certain operations on data structures more efficiently. These include:

- dynamic allocation and deallocation of software stack space when entering and leaving subroutines
- function pointer invocation
- software Stack Pointer manipulation
- manipulation of variables located in a software stack

A summary of the instructions in the extended instruction set is provided in Table 35-3. Detailed descriptions are provided in **Section 35.2.2 "Extended Instruction Set"**. The opcode field descriptions in Table 35-1 apply to both the standard and extended PIC18 instruction sets.

Note:	The instruction set extension and the Indexed Literal Offset Addressing mode were designed for optimizing applications written in C; the user may likely never use these instructions directly in assembler. The syntax for these commands is provided as a reference for users who may be reviewing code that has been generated by a compiler.

35.2.1 EXTENDED INSTRUCTION SYNTAX

Most of the extended instructions use indexed arguments, using one of the File Select Registers and some offset to specify a source or destination register. When an argument for an instruction serves as part of indexed addressing, it is enclosed in square brackets ("[ ]"). This is done to indicate that the argument is used as an index or offset. MPASM™ Assembler will flag an error if it determines that an index or offset value is not bracketed.

When the extended instruction set is enabled, brackets are also used to indicate index arguments in byte-oriented and bit-oriented instructions. This is in addition to other changes in their syntax. For more details, see **Section 35.2.3.1 "Extended Instruction Syntax with Standard PIC18 Commands"**.

Note:	In the past, square brackets have been used to denote optional arguments in the PIC18 and earlier instruction sets. In this text and going forward, optional arguments are denoted by braces ("{ }").

TABLE 35-3: EXTENSIONS TO THE PIC18 INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected
				MSb			LSb	
ADDFSR	f, k	Add literal to FSR	1	1110	1000	ffkk	kkkk	None
ADDULNK	k	Add literal to FSR2 and return	2	1110	1000	11kk	kkkk	None
CALLW		Call subroutine using WREG	2	0000	0000	0001	0100	None
MOVSF	z_s, f_d	Move z_s (source) to 1st word	2	1110	1011	0zzz	zzzz	None
		f_d (destination) 2nd word		1111	ffff	ffff	ffff	
MOVSS	z_s, z_d	Move z_s (source) to 1st word	2	1110	1011	1zzz	zzzz	None
		z_d (destination) 2nd word		1111	xxxx	xzzz	zzzz	
PUSHL	k	Store literal at FSR2, decrement FSR2	1	1110	1010	kkkk	kkkk	None
SUBFSR	f, k	Subtract literal from FSR	1	1110	1001	ffkk	kkkk	None
SUBULNK	k	Subtract literal from FSR2 and return	2	1110	1001	11kk	kkkk	None

TABLE 37-14: ANALOG-TO-DIGITAL CONVERTER (ADC) CONVERSION TIMING SPECIFICATIONS

Standard Operating Conditions (unless otherwise stated)							
Param No.	**Sym.**	**Characteristic**	**Min.**	**Typ†**	**Max.**	**Units**	**Conditions**
AD20	TAD	ADC Clock Period	1	—	9	µs	Using FOSC as the ADC clock source ADOCS = 0
AD21			—	2	—	µs	Using FRC as the ADC clock source ADOCS = 1
AD22	TCNV	Conversion Time(1)	—	11 + 3TCY	—	TAD	Set of GO/$\overline{\text{DONE}}$ bit to Clear of GO/$\overline{\text{DONE}}$ bit
AD23	TACQ	Acquisition Time	—	2	—	µs	
AD24	THCD	Sample and Hold Capacitor Disconnect Time	—	—	—	µs	FOSC-based clock source FRC-based clock source

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: Does not apply for the ADCRC oscillator.

FIGURE 37-10: ADC CONVERSION TIMING (ADC CLOCK FOSC-BASED)